한국 CAD/CAM 학회 논문집
제 13권 제 6호 2008년 12월 pp. 450-458

기술논문

삼차원 직교 격자 생성을 위한 단면 커브를 이용한 옥트리 생성과 셀 절단 알고리듬

김동훈*, 신하용**, 박세연*, 이일랑*, 권장혁***, 권오준***

Octree Generation and Clipping Algorithm using Section Curves for Three Dimensional Cartesian Grid Generation

Donghun Kim*, Hayong Shin**, Seyoun Park*, Il-lang Yi*, Janghyuk Kwon***, Oh Joon Kwon***

ABSTRACT

Recently, Cartesian grid approach has been popular to generate grid meshes for complex geometries in CFD (Computational Fluid Dynamics) because it is based on the non-body-fitted technique. This paper presents a method of an octree generation and boundary cell clipping using section curves for fast octree generation and elimination of redundant intersections between boundary cells and triangles from 3D triangular mesh. The proposed octree generation method uses 2D Scan-Converting line algorithm, and the clipping is done by parameterization of vertices from section curves. Experimental results provide octree generation time as well as Cut-cell clipping time of several models. The result shows that the proposed octree generation is fast and has linear relationship between grid generation time and the number of cut-cells.

Key words : Cartesian Grid, Octree Generation, Clipping, Section Curve, CFD

1. 서 론

효율적인 전산유체역학(Computational Fluid Dynamics, 이하 CFD) 해석을 위해서는 유동(fluid)을 표현하기 위해 공간을 유한 체적으로 나누어 사용하며, 격자의 형태가 해의 정확도에 많은 영향을 끼치게 되어, 정확한 격자 생성이 필수적이다. 그러나 복잡한 형상의 물체에 대한 격자 생성에는 많은 시간과 노력이 소요된다[10]. 이에 따라 기존에 격자 생성 시간을 획기적으로 단축하기 위해서 다양한 연구가 진행 되어 왔다[16]. 이 방식들은 크게 비 정렬 방식과 정렬 방식, 그리고 직교격자 방식으로 나눌 수 있다. 본 연구에서 대상으로 하는 직교격자방식은 크게는 *non-body-fitted* 방식으로, 격자를 표면 형상과 분리(decoupling)함으로써 복잡한 형상에 대해서도 격자 생성을 자동화 할 수 있으며, 최소한의 cell과 face를 사용하여 공간을 효율적으로 채울 수 있다는 것이 장점이나[16] 국내에서는 연구가 미미하게 진행되었다.

CFD 해석에서 물체의 형상이 정확하게 반영되기 위해서는 물체(solid)의 경계곡면이 지나가는 셀들을 모두 찾아서 최대 단계(가장 작은 크기를 가지는 셀)까지 분할하고, 경계곡면이 지나가는 모양에 따라 절단해야 한다. 직교 격자에서 solid boundary cell(이하 boundary cell)은 물체 표면에 의해 물체 내부와 유동장을 포함한 외부 영역으로 절단되기 때문에, 본 논문에서는 *Cut-cell*이라 지칭한다. Cut-cell은 표면 형상에 의해 임의로 절단되기 때문에, 그 모습은 다양하고 복잡한 형태를 가질 수 있으며, Cut-cell을 정확히 계산하고 저장하는 것은 직교 격자 생성에서 가장 까다로운 부분이다. 한편, Cut-cell은 다수의 외부 영역으로 절단될 수 있는데, 이런 경우에 *Split*이 발생했다고 한다. Berger[4]에 따르면, 비행체의 얇은 날개 부분에서 생성될 수 있는 *Split cell*을 정확히 발견하고 분석을 위해 연결 관계를 구성하는 일은 어려운 일로 남아

*학생회원, KAIST 산업및시스템공학과
**책임저자, 종신회원, KAIST 산업및시스템공학과
***비회원, KAIST 항공우주공학전공
- 논문투고일: 2008. 03. 21
- 논문수정일: 2008. 10. 24
- 심사완료일: 2008. 10. 28

있다.

직교 격자 생성 방법 가운데, 공간 정보를 효율적으로 저장하기 위해 널리 사용되는 방법은 Octree/Quad tree 등의 tree 구조를 사용하는 것이다. Octree 구조는 tree 탐색을 통한 격자 간의 연결정보를 제공하고, 각 노드로부터 격자 위치 정보를 쉽게 얻을 수 있다는 장점을 가진다[1]. 본 논문에서는 삼차원 삼각망(triangular net)으로 구성된 물체에 대한 직교 격자 생성을 위해 octree 구조를 사용하였으며, 효율적인 octree 생성과 Cut-cell 처리 방법을 제시한다.

2. 관련 연구

2.1 직교 격자 관련 기존 연구

기존에 직교격자를 이용한 CFD 해석에 많은 연구를 수행한 Aftosmis[11]와 Hunt[10]는, 직교 격자 생성을 위해서 각 축의 방향으로 등일한 크기의 셀들을 생성하고 형상 정보를 토대로 이를 분할하였다. Cut-cell의 절단을 위해, Sutherland-Hodgman[14]가 제안한 polygon-clipping algorithm 을 사용하고 있다. 그러나 이러한 방식은 셀 분할을 위해서 표면 형상과 셀간의 교차 테스트가 필요하기 때문에 octree 생성에 많은 시간이 필요하다. 한편, 삼각형이 두 개 이상의 cell과 접해 있는 경우, 삼각형-셀 절단 시 인접 셀에 대해서 중복 계산이 일어난다. 또, Cut-cell이 *split* 되는 경우를 효과적으로 처리하지 못하였다[10]. 또한 Charlton[7]은 polygonal mesh 형태의 입력을 받아 octree 구조를 사용하여 직교 격자를 생성하였으나, 구체적인 Cut-cell clipping 방법은 제시하지 못하였다.

2.2 Octree 관련 기존 연구

반면 컴퓨터 그래픽스 분야에서는 Boundary(surface) 모델을 octree 구조로 변환하는 알고리듬은 80년대부터 활발히 연구되어 오며[12][13], 주로3D range data, 혹은 silhouette image에 대한 octree 근사에 치중되어 왔다[9]. Navazo[15] 등은 octree 저장 공간을 줄이기 위해, leaf node를 face, edge, vertex node로 구분하여, 최대분할이 곡면의 내부에서는 이루어 지지 않도록 하는 방식을 사용하기도 하였다.

최근에는 대용량 polygonal mesh 데이터의 간략화(Simplification)나 mesh model의 수정 등에 octree 구조를 사용한 논문들이 발표되었다[2][3]. 그 중 Ju[11]는 잡음이 존재하는 불완전한 삼각망 모델 복원을 위해 octree 구조를 사용하였다. Incremental octree 생성 알고리듬을 사용하여 메모리 사용을 줄였으며, 등간격의 격자를 생성한 후 Separating Axis method를 사용하여 삼각형-셀간 교차 여부를 판단하여 cell을 분할하였다.

한편, 컴퓨터 그래픽스 분야의 scan-converting 3D polygon algorithm[12]은 삼차원 상에서 다각형을 voxel로 근사하는데 효율적이고 빠른 방법이다. 하지만 이 방법은 근사적인 방법이므로 본 연구에서 대상으로 하는 직교 격자에 적용하기 위해서는 약간의 수정이 필요하다.

3. 연구 범위 및 전반적 절차

본 논문의 연구 범위는 다음과 같다.

Input: 삼각망으로 이루어 진 닫혀진 물체(solid) 표면 형상, octree의 최대 분할 단계, 전체 유동장을 포함하는 외곽 boundary size

Output: 물체 외부의 공간을 포함하는 비 점성(inviscid) 직교 격자

본 논문에서 제안하는 직교 격자 생성 단계는 다음

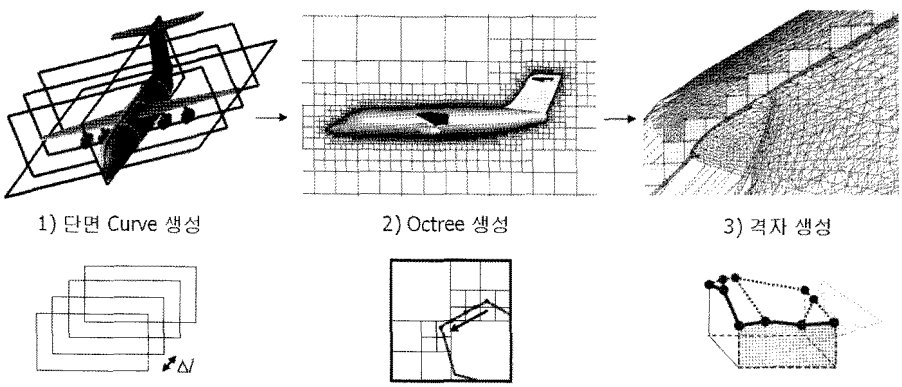


Fig. 1. 전반적 절차

과 같이 크게 3단계로 나누어 지며(Fig. 1), 각각에 대한 자세한 내용은 4, 5장에 설명되어 있다.

1. 단면 곡선 생성: 각 축의 방향(X, Y, Z)에 대해 최대 분할 단계에 의해 계산된 단면의 수만큼 단면을 생성하고, 각 단면과 삼각망이 교차하여 생기는 edge 들로 단면 곡선을 생성한다.
2. Octree 생성: 2D Scan-Converting line algorithm을 이용하여 단면 곡선을 따라가며 cell을 분할한다. 삼각망의 꼭지점 정보를 사용하여 sharp vertex를 포함한 cell이 boundary cell이 되도록 분할한다.
3. 격자 생성: boundary cell 에 대해 단면 곡선을 재 사용하여 절단 형상을 계산한다. 이 단계는 다시 *cell face* 생성과 Patch 생성으로 나뉜다. 최종적으로, Cut-cell 토폴로지(Topology)를 구성하여 직교 격자를 생성한다.

4. Octree Generation

Octree는 크게 top-down 방식과 bottom-up 방식에 의해 생성될 수 있다. Top-down 방식은 하나의 셀로 시작해서, 삼각망이 셀을 지나가는 지를 테스트하고, 지나가는 경우 분할해 나가는 재귀적인 방법이다. Bottom-up 방식은, octree를 먼저 최대 분할단계까지 uniform grid로 분할해 놓고, 삼각망이 지나가는 셀을 찾은 후, 그렇지 않은 셀들을 합쳐나가는 방식이다. 삼차원 직교 격자 생성에 bottom-up 방식을 사용하게 되면, 격자 초기화 시 최대 분할 단계가 10 이상인 경우 수 기가바이트 이상의 메모리를 필요로 하게 되어, 일반 PC에서는 생성하기 어려운 문제가 있다. 또한 top-down 방식은 한꺼번에 많은 메모리를 필요로 하지는 않으나, 매 분할 단계마다 셀-삼각형 간 교차 테스트 수행을 해야 하므로 중복된 계산으로 인해, octree 생성 시 많은 시간이 걸리게 된다.

본 연구에서는 생성시간이 빠르면서도 메모리 측면에서도 일반 PC에서 수행 가능한 격자생성을 위해, 두 방식의 장점만을 취하였다. 메모리 절약을 위해 top-down 방식에 기초하여 octree를 생성하되, Fig 1.1과 같이 미리 최대 분할단계에서의 셀 크기인 minimum length Δl을 계산하여 Δl 간격으로 단면을 생성하고, 단면 곡선과 교차하는 셀들만을 최대 단계까지 분할하였다. 이러한 접근 방법을 통해서 top-down 방식과 bottom-up 방식의 장점을 합쳐 메모리 사용을 줄이고 생성시간을 빠르게 할 수 있음을 확인하였다.

4.1 단면 곡선의 정의

앞서 설명한 바와 같이, octree 생성을 위해 본 연구에서는 삼각망과의 교차선을 저장하기 위한 단면들을 생성하며, 각 단면은 전체 좌표계의 (x, y, z) 축의 방향과 수직한 평면이다. 단면곡선들은 각 축을 따라 일정한 거리 간격으로 생성되며, 그 간격은 octree 에서 최대 분할 단계 셀의 길이 Δl로 정한다. 입력 모델은 닫힌(2d manifold) 곡면으로 가정하고 있으므로 각 단면마다 삼각망의 교차에 의해 Fig. 2와 같이 닫힌 2D polygon이 얻어진다. 한 단면당 2개 이상의 분리된 polygon이 얻어질 수도 있으며, 모든 polygon들은 반시계방향을 따라 edge들을 저장해 둔다.

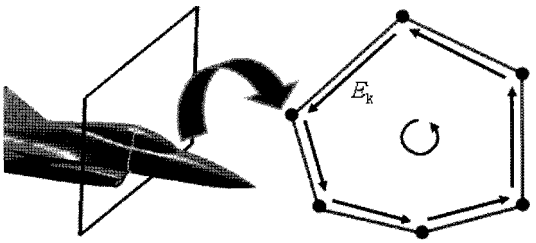


Fig. 2. 단면 곡선; axis-aligned 평면에서 삼각망과의 교차곡선들을 계산하여 저장

4.2 Octree generation

각 단면곡선들을 얻고 나면, 이를 이용하여 손쉽게 octree를 만들 수 있다. 본 연구에서 제안하는 octree 구조에서 leaf cell은 다음과 같이 3가지 type으로 분류된다.

- **boundary cell**: 물체 표면과 교차하는 셀
- **inside cell**: 물체 내부에 존재하는 셀
- **outside cell**: 유동장을 포함한 물체 외부에 존재하는 셀

Boundary cell들을 잘 분류하고 나면 in/out을 구분하는 일은 상대적으로 쉬운 일이다. 따라서 먼저 boundary cell들을 찾아 최대 단계까지 나누는 일을 4.2.1에 설명된 2D scan conversion 방식을 따라 수행한다.

4.2.1 2D scan conversion

한 단면 곡선 edge E_k가 관통하는 셀들을 빠뜨리지 않고 최대 level로 분할하기 위해 다음과 같이 Bresenham[6]의 2D scan-converting line 알고리즘을 수정하여 사용하였다.

한 단면 곡선이 m개의 edge로 구성되었다고 할 때, 단면 곡선 edge E_k (k=1,..., m)에 대해, E_k를 이루는 두

vertex 중, 시작 vertex는 SP_k, 끝 vertex를 EP_k라 하자. 정의된 edge와 vertex에 의해 2D Scan-Converting line 알고리듬은 다음과 같이 수행된다.

1) SP_1를 포함하는 셀을 최대 단계까지 분할한다.
2) 각 E_k (k=1,..., m)에 대해 다음을 수행한다.
a) EP_k를 포함하는 cell을 최대 level까지 분할한다.
b) SP_k부터 시작하여 EP_k까지, E_k를 따라 이동하면서 교차하는 모든 셀들을 최대 단계까지 쪼갠다.
c) E_k가 놓여있는 boundary cell 내부 데이터 구조에 E_k를 저장해둔다. (이는 향후 clipping에서 사용하기 위함이다.)

4.2.2 추가 작업

Fig. 3과 같이 sharp vertex를 포함한 셀의 경우, 셀을 지나는 단면 곡선이 셀의 한 변에만 놓이게 된다. 또는 셀의 크기가 삼각형의 크기에 비해 상대적으로 큰 경우 단면 사이에 들어오는 삼각형들이 존재할 수 있으며, 이러한 삼각형들이 포함된 셀은 단면곡선만으로는 찾아지지 않을 수 있다. 따라서 삼각형 vertex들만을 이용하여, 각 vertex를 포함하는 셀을 모두 최대 단계까지 분할 하는 과정을 한 번 더 거쳐야만 정확한 boundary cell들을 모두 찾아낼 수 있다.

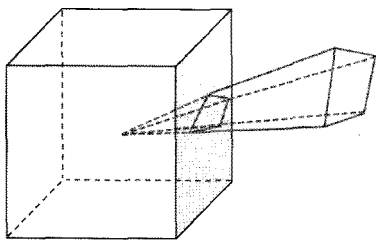


Fig. 3. Sharp vertex; vertex를 포함하는 셀은 단면곡선만으로는 찾아지지 않을 수 있다.

한편, 좀 더 좋은 해석 결과를 얻기 위해서는 balanced octree가 필요하여 balancing을 수행한다. Octree leaf cell들과 인접한 셀의 단계가 1이상 차이가 나는 경우에 단계의 차이를 1 이하로 조정하기 위해 balancing이 수행된다. 마지막으로, 유동 해석을 위해서는 유동이 존재하는 셀의 형상만이 필요하고 solid 내부의 셀에 대한 정보는 필요가 없다. 따라서 Cut-cell이 표면형상에 의해 절단되기 전에 입력 곡면의 안/바깥쪽 셀의 구분을 하여 유동에 포함되는 부분만을 추출한다.

5. Cut-cell 절단

직교 격자 생성에서 Cut-cell의 절단 형상을 정확히 계산하고 저장하는 것이 중요하다. Cut-cell을 기준으로 삼각형과 절단(clipping) 수행 시, 삼각형과 교차하는 인접 cell에서 *cell face*가 중복 계산된다. 이를 확장시켜 보면, 인접하고 있는 boundary cell 전체에 대해서 중복 계산이 발생한다. 중복 계산을 방지하기 위해 본 연구에서는 Fig. 4에서 보이는 바와 같이 앞서 계산한 단면 곡선을 사용한 clipping 방식을 제안한다.

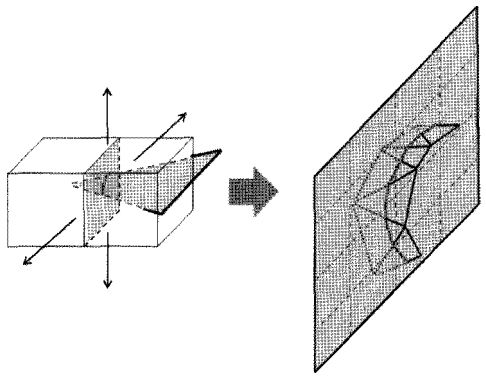


Fig. 4. 단면 곡선을 이용한 인접 Cell 중복 계산 제거

5.1 Cut-cell 절단 형상 정의

Octree cell과 삼각형 모두 convex이므로, Cut-cell과 삼각형의 교차는 convex polygon을 형성한다. Cut-cell은 6개 면(f_1, ..., f_6)에 대해서, 적어도 하나 이상의 면이 단면 곡선과 교차한다. Cut-cell과 교차하는 삼각형을 Cut-cell 내부에 들어오는 부분만 남기고, Fig. 5와 같이 잘라내는 과정을 본 논문에서는 3D clipping이라 부르며, Cut-cell 절단 시 필요한 용어들을 다음과 같이 정의한다.

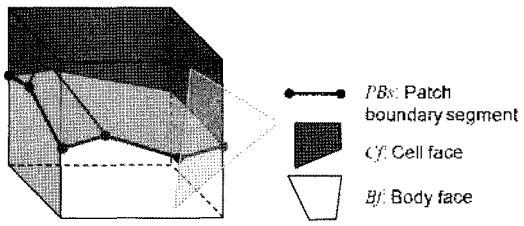


Fig. 5. Cut-cell 절단 형상 정의

Definition 1. 한 Cut-cell내에 삼각형들에 의해 새로 생기는 절단면을, solid body에 의해 생성되므로 *body face* Bf_i 라고 정의한다. 이는 항상 convex 형상

을 띄며, h개의 triangle이 Cut-cell과 교차한다면, 한 cell내에서 body face들의 집합인 patch P는 $P=\{Bf_i | i=1, ..., h\}$로 정의할 수 있다.

Definition 2. *Patch boundary* 집합 PB는 Cut-cell의 face에서 찾아진 단면 곡선 edge E_k들에 의해 구성된다. 이 때, E_k의 일부분만 Cut-cell의 면 상에 놓이게 되는데, 이 선분들을 patch boundary segment PBs_i라고 정의한다. t개의 PBs_i가 한 Cut-cell에 존재하면, $PB=\{PBs_i | i=1, ..., t\}$로 정의된다.

Definition 3. 단면 곡선이 Cut-cell의 face f_j를 지나갈 때, 단면 곡선에 의해 f_j는 두 개 이상의 polygon으로 나눠진다. 이 과정을 2D Clipping이라 하자. Cut-cell의 각 face f_j를 단면 곡선에 대해서, 2D clipping하여 얻은 polygon을 *cell face* Cf_i라고 정의한다.

Cut-cell size는 최대 분할 단계가 커짐에 따라, 충분히 물체의 local feature size보다 작기 때문에 물체가 한 Cut-cell을 두 번 이상 관통하여 Cut-cell이 두 개 이상의 내부 영역을 가지는 경우는 발생하지 않는다고 가정한다. 또한, 3D clipping과 2D clipping을 통해서 유동장을 포함한 외부 영역만을 저장한다고 가정한다.

5.2 Cell-face 생성

단면 곡선과 Cut-cell의 face $f_j(j=1, ..., 6)$ 사이의 2D clipping을 통해 *cell face*가 얻어진다. 먼저, f_j에 놓여진 단면 곡선 edge E_k를 셀에 저장된 edge list에서 찾아온다. 그리고 E_k와 f_j가 만나는 형태에 따라 4가지 type으로 구분하여 2D clipping을 수행한다.

2D clipping 단계는 *cell face*를 구성하는 vertex들을 CFVL(Cell Face Vertex List)에 parameter 값에 따라 오름차순으로 저장하는 방식으로 진행된다. Fig. 6에 나타낸 것과 같이, 4가지 type의 구분과 2D clipping 방식은 다음과 같다.

1) **type-1**: 하나의 E_k가 f_j와 교차하는 경우. E_k가 f_j와 교차하여 생긴 교점 두 개를 P, Q라 하자. P, Q의 parameter 값 들과 E_k의 방향을 고려하여 starting vertex와 ending vertex를 지정한다. Starting vertex을 *sv*, ending vertex을 *ev*라고 부르자. 우선, *sv*를 CFVL에 저장하고, 반 시계 방향으로 f_j의 edge를 따라 *sv*와 *ev*사이에 있는 vertex(v_i)들을 CFVL에 추가한다. 마지막으로 *ev*를 CFVL에 추가한다(Fig. 6(a)).
2) **type-2**: 두 개 이상의 연결된 E_k가 f_j와 교차하는 경우. type-1에서와 마찬가지로, starting vertex *sv*, ending vertex *ev*를 정한다. type-2의 경우, E_k 들 중에 f_j 내부에 완전히 포함되는 edge들부터 먼저 시계 반대방향으로 CFVL에 저장한다. 그 후에는 type-1과 마찬가지로 *sv*, *sv*와 *ev* 사이에 존재하는 vertex(v_i), *ev*를 차례로 CFVL에 저장한다(Fig. 6(b)).
3) **type-3**: type-3의 경우는 f_j에 2개의 외부영역이 존재하게 된다. 이 경우 *Split* 이 발생할 수 있다. 연결된 edge들로 type-1 혹은 type-2의 방식으로 처리한 후, 2D clipping 되고 난 나머지 영역의 vertex parameter 값들을 저장해 두고, 이를 다시 type-1 혹은 type-2 방식으로 처리하도록 한다(Fig. 6(c)).
4) **type-4**: 이 경우, 단면 곡선의 모든 edge들이 Cut-cell의 면 f_j에 포함된다. 즉, 단면 곡선으로 둘러싸인 내부 영역과 f_j에서 단면 곡선을 제외한 외부 영역으로 나눌 수 있는데, 이 외부 영역은 홀(hole)이 있는 사각형이므로, 이런 경우에 두 개의 다각형으로 쪼개도록 한다(Fig. 6(d)).

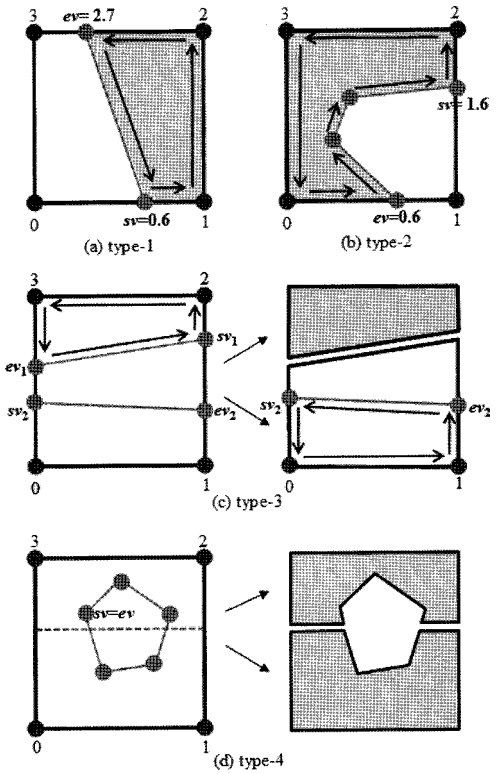


Fig. 6. 교차유형별(type) 2D clipping; (a)하나의 edge가 셀을 지나는 경우; (b)두 개 이상의 연결된edge가 셀을 지나는 경우; (c)셀이 edge들에 의해 3개 이상으로 분리되는 경우; (d)다각형이 셀 내부에 포함되는 경우

Type-4를 제외하면 본 연구에서 사용하는 2D clipping algorithm은 Weiler-Atherton algorithm[8]과 동일하다.

5.3 Patch 생성

Patch는 body face들로 구성되며, body face는 Cut-cell과 삼각형의 3D clipping에 의해 생성된다. *Cell face* 생성단계에서 계산한 Patch boundary segment PBs_i 정보를 활용하고 삼각형 꼭지점이 셀 내부에 들어오는 경우를 고려하여 body face를 생성하는 방법을 제안한다. BFVL는 body face를 구성하는 vertex들을 저장하는 자료구조라고 생각하자. Patch **P**를 생성하는 과정은 다음과 같다.

1) PBs_i 정보와 연결 삼각형 정보를 가지고, PBs_i의 양 끝 vertex들의 삼각형 내 파라미터 값을 계산한다.
2) vertex의 파라미터 값의 순서대로 vertex를 해당 BFVL에 추가한다.
3) Cut-cell과 교차하는 삼각형들 중에서, 삼각형 vertex가 Cut-cell의 내부에 위치하면 이 vertex도 BFVL 에 추가한다.
4) Cut-cell의 6개 면 (f_1, ..., f_6)에 대해서 위의 과정을 반복한다.

단면 곡선의 edge E_k는 자신을 포함하는 삼각형을 참조하여 저장하도록 되어 있고, 이 삼각형을 T_i라고 하자. PBs_i가 단면 곡선 edge E_k에서 찾아 졌다면, PBs_i의 양 끝 vertex들은 T_i 내부에 속한다. 한편, V를 PBs_i의 양 끝 vertex 중 하나라고 하고 E_k가 삼각형 T_i와 만나는 edge의 시작 vertex를 sv, 끝 vertex를 ev라 하자. P_V를 V의 parameter 값이라고 할 때, P_V는 sv, ev, V로 계산할 수 있다. $\mathbf{d} = \mathbf{r}(ev) - \mathbf{r}(sv)$로 정의한다. ($\mathbf{r}(v)$는 v의 위치 벡터로 정의한다.) 그러면 V의 parameter 값 P_V는 식 (1)과 같이 정의된다(Fig. 7 참조).

$$P_V = \frac{(\mathbf{r}(V) - \mathbf{r}(sv)) \cdot \mathbf{d}}{\|\mathbf{d}\|^2} + P_{sv} \qquad (1)$$

각각의 PBs_i의 양 끝 vertex에 대해 파라미터 값을 계산하고, 그 순서대로 BFVL에 저장한다. Cut-cell 내부에 위치한 삼각형 vertex의 경우, vertex 번호(0, 1, 2)를 파라미터 값으로 하여 BFVL에 저장한다. 삼각형 세 꼭지점이 모두 Cut-cell 내부에 위치할 경우, 이 삼각형은 스스로 body face를 구성하므로, 삼각형 무게중심 값을 검사하여 patch에 추가하도록 한다. 한 Cut-cell과 다수의 삼각형이 교차하면 BFVL도 여러 개가 생성된다.

지금까지 *cell face*와 patch 생성 방법을 살펴 보았다. 파라미터 값 순서대로 vertex들을 저장함으로써, 불필요한 sorting을 없앨 수 있다. 특히 Cut-cell과 삼각형 간 교차 type을 구분할 필요 없이 Body face를 생성할 수 있고, *cell face*의 유형구분을 통해서 *Split* 되는 경우도 쉽게 처리할 수 있다.

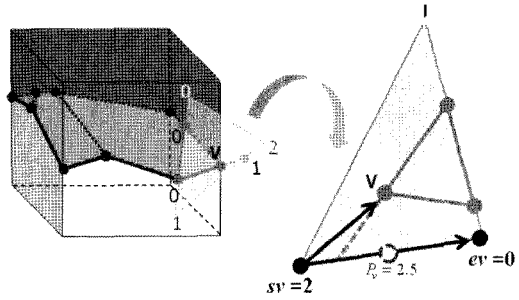


Fig. 7. Parameterization of V

5.4 격자 생성

Cut-cell의 절단면들을 모두 계산하고 필요한 vertex와 면을 저장하고 나면 격자를 생성할 수 있다. 유동장을 포함한 외부 영역을 분석 영역으로 선택하고, Cut-cell 계산과정에서는 외부 영역의 body face, *cell face*만을 저장한다. 이 때, body face와 *cell face*는 동일한 face 요소로 생각한다. *Split Cell*의 경우, 차후 계산에서는 boundary cell이 두 개의 외부영역을 소유하고 있다는 정보를 알고 있지만, 격자 생성 뒤에는 나눠진 외부 영역끼리의 연결 정보는 사라진다.

6. 결과 및 분석

본 연구에서 제안하는 방법을 사용하여 Octree를 생성하고 Cut-cell clipping을 수행하였고, octree 생성 시간과 clipping 시간을 측정하였다. 제안된 알고리즘은 Microsoft Windows XP OS에서 Microsoft Visual C++.NET 2003 (ver. 7.1) 환경으로 구현되었고, Intel Pentium 4 processor (3.0 GHz), 3.0 GB RAM을 가진 PC에서 테스트 하였다.

6.1 Octree 생성과 셀 절단 수행 결과

Fig. 8과 Fig. 9은 각각 BAe 146(별칭 Jumbolino) 삼각망 모델과 F16 삼각망 모델을 입력모델로 사용하여 최대 분할 단계 11에서 octree를 생성한 결과 그림이다.

Fig. 10과 Fig. 11은 위 모델들에 대해, clipping을

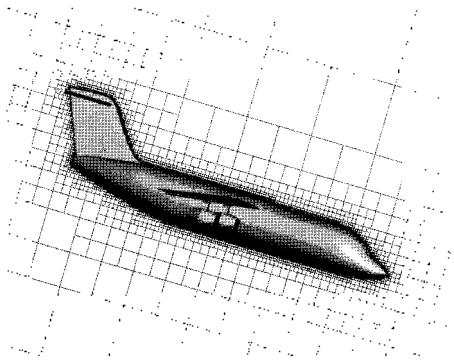


Fig. 8. Octree 생성 결과 Jumbolino model

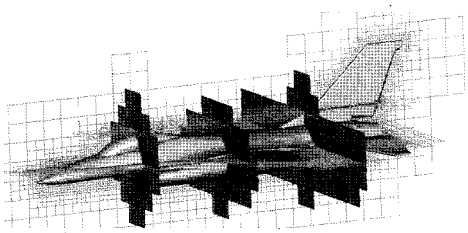


Fig. 9. Octree 생성 결과 F16 model

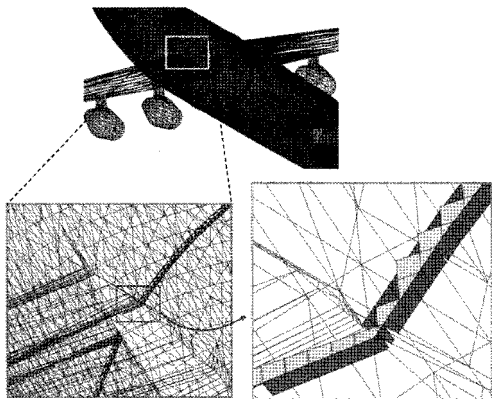


Fig. 10. Clipping 결과, Jumbolino(17만 triangles)

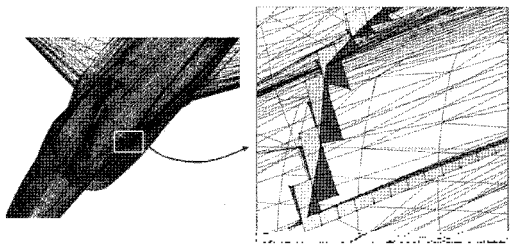


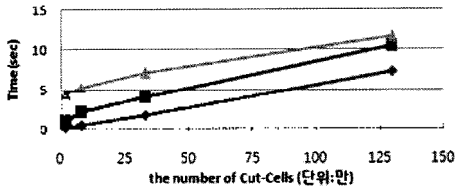
Fig. 11. Clipping 결과, F16(21만 triangles)

수행한 결과이다.

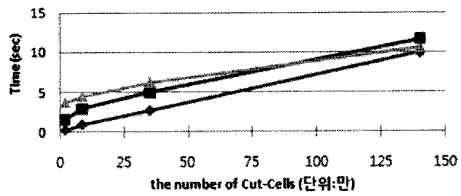
확대된 결과는 특정 단면 상의 *cell face*와 해당 *cell face*에 연결된 body face를 boundary cell 한 층에 대해서 나타낸 것이다.

6.2 Octree 생성 시간과 셀 절단 수행 시간

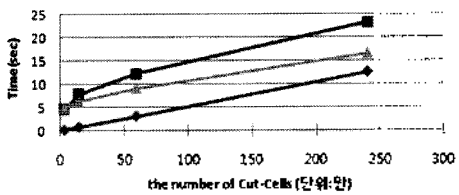
본 논문에서 제안한 2D scan-conversion에 의한 octree 생성 시간을 가장 빠르다고 알려진 octree 생성 방법인 3D scan-conversion[13] 방식과 Ju[12]가 제안한 octree 생성 방법과 비교하여 나타내었다.



(a) F16 (214,046 triangles)



(b) Jumbolino (173,049 triangles)



(c) Missile (237,884 triangles)

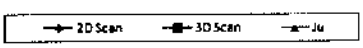


Fig. 12. Octree 생성시간 비교

시간 측정은 6.1절에서 사용한 F16 model, Jumbolino model과 간단한 missile model을 subdivision하여 삼각형 개수를 변화시켜가면서 수행하였다. 단면 곡선은 clipping 시에도 필수적으로 생성되어야 하므로, octree 생성 시간에 단면 곡선 생성시간을 포함하지 않았다.

Fig. 12과 같이 2D scan-Cconverting line algorithm을 이용한 octree 생성 방식이 다른 방식들에 비해 가장 시간이 적게 걸리는 것을 확인 할 수 있었다. 각 model들의 삼각형 개수가 40만, 60만개일 때에도 비슷한 결과를 얻을 수 있었다. 그리고 제안하는 octree

생성 방법의 소요 시간은 Cut-cell의 개수와 선형적인 관계를 나타내는 것을 확인하였다.

위에서 사용한 Jumbolino 모델과 F16 모델에 대해 Cut-cell clipping 수행 시간을 측정하였다. Clipping 수행시간은 격자 생성을 위한 Cut-cell topology 구성 시간을 포함한다. Clipping 수행 시간도 octree 생성시간과 마찬가지로 Fig. 13에서 보이는 바와 같이 Cut-cell 개수와 선형적 관계로 나타나는 것을 알 수 있다.

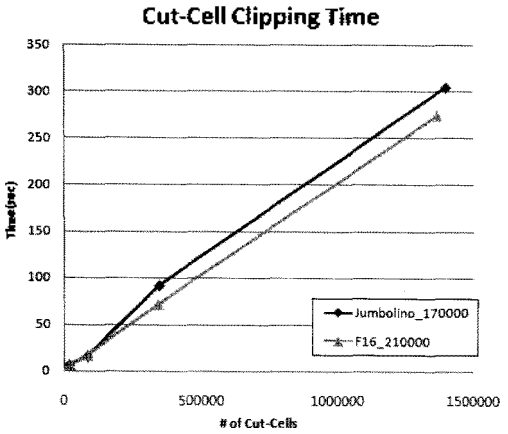


Fig. 13. 셀 절단 수행시간

7. 결론 및 추후 연구

본 연구에서는 임의의 삼각망 모델에 대한 CFD 분석을 위한 삼차원 적교 격자 생성 방법을 제안했다. 먼저, 각 축의 방향으로 단면을 생성하여, 단면 곡선들을 정의하였고, 이 단면 곡선을 사용하여 octree 생성과 Cut-cell clipping을 수행하였다. 제안하는 방법은 octree 생성을 빠르게 하고 Cut-cell clipping의 중복 계산을 줄일 수 있었다.

6장에서 제시한 결과와 같이, octree 생성 시간과 Cut-cell clipping 수행 시간은 Cut-cell의 수가 증가함에 따라, 선형적인 관계를 나타내었다. 최대 분할 단계 11에서 모델들을 clipping 한 후, Cut-cell 개수는 약 140만 개였다. 이 결과는 유동 해석을 하기 위한 격자 수로 적합하며, 구현된 프로그램을 통해 격자 생성을 충분히 자동화 할 수 있을 것으로 기대된다.

감사의 글

본 연구는 방위사업청과 국방과학연구소의 지원으로 수행되었습니다(UD080042AD).

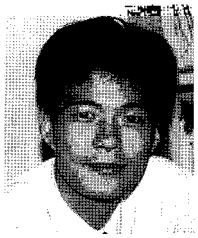
참고문헌

1. M. J. Aftosmis, "Solution adaptive Cartesian grid methods for aerodynamic flows with complex geometries," *Computational Fluid Dynamics VKI Lecture Series*. Belgium, 1997.
2. C. Andùjar, P. Brunet, and D. Ayala, "Topology-Reducing Surface Simplification Using a Discrete Solid Representation," *ACM Transactions on Graphics*, Vol. 21, No. 2, pp. 88-105, 2002.
3. D. Ayala, P. Brunet, P. Juan, and I. Navazo, "Object Representation by Means of Nonminimal Division Quadtrees and Octrees," *ACM Transactions on Graphics*, Vol. 4, No.1, pp. 41-59, 1985.
4. M. J. Berger and M. J. Aftosmis, "Aspects (and aspect ratios) of Cartesian Mesh Methods," *Proceedings of the 16th International Conf. on Num. Meth. in Fluid Dynamics*, Jul 1998.
5. J. Bonet and J. Peraire, "An alternating digital tree (ADT) Algorithm for Geometric Searching and Intersection Problems," *Int. J. Num. Meth. Eng*, Vol. 31, pp. 1-17, 1991.
6. J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems journal*, Vol. 4, No. 1, pp. 25-30. 1965.
7. E. F. Charlton, "An octree solution to conservation-laws over arbitrary regions (OSCAR) with applications to aircraft aerodynamics," Ph.D. Thesis, Univ. of Mich., Dept. of Aero. and Astro. Engr., 1997.
8. J. D. Foley, A. van Dam, S. K. Feiner, and H. F. Hughes, "Computer Graphics: Principles and Practice," *Addison Wesley*, 1996.
9. H. C. Homer and S. H. Thomas, "A Survey of Construction and Manipulation of Octrees," *Computer Vision, Graphics, and Image Processing*, Vol. 43, pp. 409-431, 1988.
10. J. D. Hunt, "An Adaptive 3D Cartesian Approach for the Parallel Computation of Inviscid Flow About Static and Dynamic Configurations," Ph.D. Thesis, Univ. of Mich., Dept. of Aero Eng and Sci Comp., 2004.
11. T. Ju, "Robust repair of polygonal models," *ACM Transactions on Graphics*. Vol. 23, No. 3, pp. 888-895, 2004.
12. A. Kaufman, "Efficient Algorithms for Scan-Converting 3D Polygons," *Comput. & Graphics*, Vol. 12, No. 2, pp. 213-219, 1988.
13. I. Navazo, D. Ayala, and P. Brunet, "A geometric modeller based on the exact octree representation of polyhedra," *Comp. Graphics Forum*. Vol. 5, No. 2, pp. 91-104, 1986.
14. I. E. Sutherland and G. W. Hodgman, "Reentrant polygon clipping," *Communications of the ACM*, Vol. 17, No. 1, pp. 32-42, 1974.

15. J. Thompson, B. Soni, and N. Weatherill, Eds, "Handbook of Grid Generation", *CRC Press*, 1998.
16. Z. J. Wang, *et al.*, "A 2^N Tree Based Automated Viscous Cartesian Grid Methodology for Feature Capturing," *AIAA Paper*, No. 99-3300, 1999.

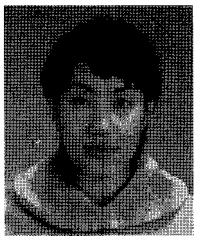
김 동 훈

2005년 KAIST 산업공학과 학사
2008년 KAIST 산업공학과 석사
2008년 2월~현재 대우조선해양 근무
관심분야: geometric modeling, computer vision



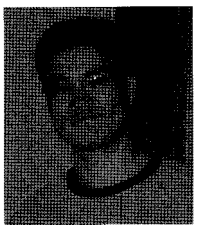
신 하 용

1985년 서울대학교 산업공학과 학사
1987년 KAIST 산업공학과 석사
1991년 KAIST 산업공학과 박사
1991년~1993년 LG생산기술원 선임연구원
1993년~1997년 (주)큐빅테크 연구소장
1997년~2001년 DaimlerChrysler Senior Specialist
2001년~현재 KAIST 산업공학과 부교수
관심분야: geometric modeling, tool path generation, process planning, computational geometry



박 세 연

2002년 2월 KAIST 산업공학과 학사
2004년 2월 KAIST 산업공학과 석사
2004년 3월~현재 KAIST 산업공학과 박사과정
관심분야: general geometric modeling, point-based modeling, computational geometry, Bio-CAD

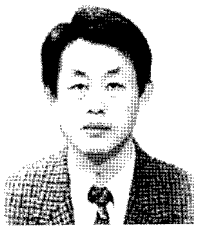


이 일 랑

2001년 한양대학교 기계공학과 학사
2003년 한양대학교 정밀기계공학과 석사
2001년~2005년 한국생산기술연구원, Engineering researcher
2006년 3월~현재 KAIST 산업공학과 박사과정
관심분야: geometric modeling, offset surface, computational geometry

권 장 혁

1971년 서울대학교 항공우주공학 학사
1977년 Mississipi 주립대학 항공우주공학 석사
1986년 Cornell University 항공우주공학 박사
1971년~1975년 공군사관학교 교관
1976년 국방과학연구소 선임연구원
1986 Cornell University Post-doc.
1987년~현재 KAIST 부교수/교수
관심분야: 전산유체역학, 3차원 형상설계, 공력형상 최적설계, 통합 설계



권 오 준

1979년 서울대학교 항공우주공학 학사
1984년 KAIST 항공우주공학 석사
1988년 Georgia Institute of Technology 항공우주공학 박사
1987년~1991년 Georgia Institute of Technology, Research Associate
1991년~1994년 NASA Lewis Research Center, Research Engineer
1994년~현재 KAIST 항공우주공학과 조교수/부교수/교수
관심분야: 전산공기역학, 멀티컴퓨터공기역학, 공력최적설계